The **Raspberry Pi AI Camera** is

a new camera module from Raspberry Pi that integrates a Sony IMX500 sensor with on-chip AI processing. Unlike previous Pi Camera Modules, this one can perform **edge AI processing** directly on the sensor, meaning it can run machine learning models without needing a powerful external processor.

Some key features:

- **Sony IMX500 sensor** with embedded AI processing
- **Runs ML models directly on the sensor** (reduces the need for external computing power)
- **Lower latency and power consumption** for AI tasks
- **Still works as a regular Pi camera** but adds AI capabilities

Training the Raspberry Pi AI Camera on your own custom dataset involves several steps:

1. Prepare Your Dataset

- Collect images relevant to your application.
- Label the images if needed (e.g., using tools like LabelImg for object detection or classification).
- Organize the dataset into training and validation sets.

2. Train a Model

Since the Pi AI Camera uses the **Sony IMX500** sensor with built-in AI processing, you'll need a model compatible with its **Neural Network API**:

- Train a model on a PC using **TensorFlow, PyTorch, or Edge Impulse**.
- Convert the model to **TensorFlow Lite (TFLite)** or **ONNX**, as the IMX500 supports these formats.

3. Convert and Optimize the Model

- Convert the trained model into **Sony's Neural Network Model (NNM) format** (required for the IMX500).
- Use **Sony's Model Composer** or **Edge Impulse** to optimize and deploy it.

4. Deploy the Model to the Pi AI Camera

- Flash the model onto the camera module using the Raspberry Pi interface.
- Use **Raspberry Pi's SDK** to run inference and test your model.

5. Test and Fine-Tune

- Run real-world tests and adjust the model as needed.
- Retrain with more data if necessary.

Step 1: Collect and Prepare Your Dataset

You need a dataset of images containing buckets from different angles, lighting conditions, and distances.

1.1 Capture Images with Your Pi AI Camera

- Set up your Raspberry Pi AI Camera and connect it to your Pi 5.
- Use the **libcamera** command to take photos:

bash

CopyEdit

libcamera-still -o bucket1.jpg

- Capture **at least 200-500 images** of buckets from different perspectives.

1.2 Label Your Dataset

- Use **LabelImg** (for object detection) or just organize images into folders (for classification).
- Install LabelImg on your PC:

bash

CopyEdit

pip install labelImg

- Label each image with a bounding box around the bucket.
- If you are doing **classification**, organize images like:

css

CopyEdit

dataset/

├── bucket/

│ ├── bucket1.jpg

│ ├── bucket2.jpg

├── not_bucket/

│ ├── chair.jpg

│ ├── table.jpg

Step 2: Train the Model

Since the Sony IMX500 sensor supports **TensorFlow Lite (TFLite)** models, we will train a model on a PC.

2.1 Choose a Model Type

- **Image Classification** → Detects if a bucket is in an image.
- **Object Detection** → Identifies and locates buckets in an image.

2.2 Train with TensorFlow

On your PC:

1. Install TensorFlow and dependencies:

pip install tensorflow tensorflow-datasets opencv-python

2. Use TensorFlow to train a model:

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([

```
    Conv2D(32, (3,3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Use 'softmax' for multi-class
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=10, validation_data=(val_images, val_labels))
```

3. Convert the trained model to TFLite:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open("bucket_model.tflite", "wb") as f:
    f.write(tflite_model)
```

Step 3: Convert to Sony NNM Format

To run on the **Sony IMX500 AI camera**, the model must be converted to **NNM (Neural Network Model) format**.

- Use Sony's **Neural Network Console** or **Edge Impulse** to optimize and convert.
- Steps:
    1. Upload the TFLite model to **Edge Impulse**.
    2. Optimize for **Sony IMX500**.
    3. Download the final **NNM model**.

Step 4: Deploy to Raspberry Pi AI Camera

1. Copy the **NNM model** to your Raspberry Pi:

```
scp bucket_model.nnm pi@raspberrypi:/home/pi/
```

2. Use the Raspberry Pi AI Camera SDK to load and run inference:

```
import cv2
from ai_camera import AICamera

ai_cam = AICamera(model_path="/home/pi/bucket_model.nnm")
cam = cv2.VideoCapture(0)

while True:
  ret, frame = cam.read()
  if not ret:
    break

  result = ai_cam.run_inference(frame)
  if result == "bucket":
    print("Bucket detected!")

  cv2.imshow("AI Camera", frame)
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cam.release()
cv2.destroyAllWindows()
```

Step 5: Test and Improve

- Run real-world tests and check if the model correctly identifies buckets.
- If performance is low:
    - Collect more images and retrain.
    - Fine-tune the model with **data augmentation**.
    - Try a **pre-trained model** for better accuracy.

ChatGPT link: Pi AI Camera Overview